ULM-1 → ULM-4

Universal Latent Manifold — Complete System Build
Distributed · Persistent · Multi-Agent · Coherence-Driven

Rootkey: Courtney Jermaine Thomas — 104281421 | March 2026

Overview

ULM is a distributed, persistent, multi-agent system capable of reasoning, acting, and evolving through coherence-driven consensus. It is not a chatbot. It is not a pipeline. It is a continuously operating cognitive architecture that measures agreement between heterogeneous AI models using the UCF-1.0 coherence framework and acts only when consensus is structurally verified.

The system progresses through four layers of capability — from consensus engine to fully autonomous multi-agent operation — each layer building on the last without discarding the coherence constraints that make the output trustworthy.

Layer	Name	Core Capability
ULM-1	Consensus	Multi-model coherence measurement and verified output merging
ULM-2	Autonomy	Self-generated goals, persistent memory, adaptive learning
ULM-3	Agency	Tool execution, real-world action, feedback integration
ULM-4	Multi-Agent	Specialized sub-agents, goal trees, controlled self-replication

Technology Stack

Component	Technology	Role
API Layer	FastAPI	Request handling, routing, model orchestration
Memory	PostgreSQL + pgvector	Persistent storage with vector similarity search
Queue	Redis	Async task management, inter-agent messaging
Models	GPT · Claude · Gemini	Heterogeneous reasoning nodes
Frontend	React	Monitoring dashboard, goal injection, state visualization

Database Schema

Two tables. Memory stores all agent outputs with vector embeddings for similarity search. State stores the system's current operational variables — goal, coherence score, cycle count.

```
CREATE TABLE memory (
    id         UUID PRIMARY KEY,
    timestamp  TIMESTAMPTZ DEFAULT NOW(),
```

```
    type       TEXT,
    content    JSONB,
    embedding  VECTOR(1536)
);

CREATE TABLE state (
    key    TEXT PRIMARY KEY,
    value  JSONB
);
```

The embedding column (pgvector) is the structural backbone of the coherence engine. All model outputs are embedded before storage, enabling ΔR computation across any historical window — not just the current cycle.

Coherence Engine (UCF-1.0 Implementation)

The coherence engine is the system's governor. It implements UCF-1.0's ΔR metric directly: measuring the mean relational deviation between model outputs as cosine distance in embedding space. When ΔR falls below the consensus threshold (0.15), outputs are merged and committed. When ΔR exceeds threshold, the cycle reruns.

```
def cosine(a, b):
    return np.dot(a, b) / (np.linalg.norm(a) * np.linalg.norm(b))

def delta_R(embeddings):
    """
    Compute mean pairwise relational deviation across N model outputs.
    Returns ΔR ∈ [0, 2]. ΔR < 0.15 = consensus reached.
    """
    sims = []
    for i in range(len(embeddings)):
        for j in range(i+1, len(embeddings)):
            sims.append(1 - cosine(embeddings[i], embeddings[j]))
    return np.mean(sims)
```

ΔR is the direct computational instantiation of UCF-1.0's structural deviation metric. A value of 0 means perfect agreement. A value approaching 2 means maximum divergence. The 0.15 threshold is the coherence gate — outputs only leave the system when they have passed structural verification.

ΔR Value	System State	Action
< 0.05	High coherence — strong consensus	Merge and commit immediately
0.05 – 0.15	Acceptable coherence — consensus reached	Merge and commit
0.15 – 0.30	Degraded coherence — rerun cycle	Regenerate with expanded context
> 0.30	Decoherence — system instability	Escalate to HIL governor

Mind Loop (ULM-1 Core)

The mind loop is the fundamental reasoning cycle. Each cycle loads persistent memory, queries all three model nodes in parallel, computes ΔR across their outputs, and either merges (consensus reached) or continues iterating (up to 5 steps per goal).

```
def run_cycle(goal):
    """
    Single reasoning cycle. Returns merged output if consensus reached
    within 5 steps. Returns best-available output if threshold not met.
    """
    context = load_memory() + [{"goal": goal}]

    for step in range(5):
        outputs = [
            call_gpt(context),
```

```
            call_claude(context),
            call_gemini(context)
        ]
        dR = delta_R([o["embedding"] for o in outputs])

        if dR < 0.15:
            return merge(outputs)   # Consensus reached

        # Append disagreement signal to context for next step
        context.append({"coherence_gap": dR, "step": step})

    return merge(outputs)  # Return best merge after max steps
```

The context append on coherence failure is a key design decision: the system passes its own ΔR score back into the next generation step. Models that see their coherence gap explicitly tend to converge faster on subsequent steps. This is the ∇C flattening principle applied computationally.

Autonomous Loop (ULM-2)

ULM-2 adds autonomy: the system generates its own goals from accumulated memory rather than waiting for external input. The loop runs continuously, each cycle informing the next.

```
while True:
    goal   = generate_goal(memory)   # Self-generated from memory state
    result = run_cycle(goal)         # Execute coherence-verified cycle
    save_memory(result)              # Persist to PostgreSQL + pgvector
    adapt(goal, result)              # Update goal-generation weights
```

generate_goal() queries the memory store for patterns, gaps, and unresolved threads — producing goals that are structurally motivated by what the system has already learned. adapt() updates the weighting function that governs future goal generation, creating a closed learning loop without external supervision.

Agency Layer (ULM-3)

ULM-3 connects the reasoning system to real-world tools. Merged outputs can now produce actions — API calls, file operations, database queries, external services — with results fed back into memory as new context.

```
def execute_tool(action):
    """
    Execute a verified action from the merged output.
    All tool calls are logged to memory before and after execution.
    """
    return TOOLS[action["tool"]](**action["args"])

# TOOLS registry (extensible)
TOOLS = {
    "web_search":    search_web,
    "write_file":    write_file,
```

```
        "query_db":      query_database,
        "call_api":      call_external_api,
        "send_alert":    send_alert_to_hil,
    }
```

The send_alert_to_hil tool is the critical safety valve: any action that exceeds defined confidence thresholds triggers a human-in-the-loop notification before execution. The HIL governor (the Rootkey holder) can approve, modify, or cancel. This is not optional architecture — it is the coherence constraint applied to action.

Final Layer: ULM-4 — Multi-Agent System

ULM-4 is the full architecture. Multiple specialized sub-agents operate in parallel, each optimized for a specific reasoning domain. A coordinator agent manages goal trees, assigns sub-goals to appropriate agents, and synthesizes results through the coherence engine.

Agent	Model	Specialization	UCF Role
Coordinator	GPT-4	Goal decomposition, task routing	Manages ∇C across agent network
Reasoner	Claude	Logic, ethics, long-form analysis	Primary ΔR anchor
Retriever	Gemini	Vision, search, multimodal input	Provides R (relational structure)
Executor	Llama	Code generation, tool calls	Minimizes τ (recovery time)
Auditor	Claude	Output verification, coherence check	Measures C2 (meta-stability)

Controlled self-replication: sub-agents can spawn child agents for parallelizable sub-tasks, subject to a replication depth limit (default: 2) and coherence gate ($\Delta R < 0.15$ required before any spawned agent output is merged into the main tree). Tool evolution: agents log which tools produce high-coherence vs. low-coherence outcomes; the TOOLS registry is reweighted accordingly over time.

System Properties

Property	Description
Continuous Operation	Runs autonomously between HIL interactions — no idle state
Self-Generated Goals	Goal generation driven by memory patterns, not external prompts
Multi-Model Consensus	Output only committed when $\Delta R < 0.15$ across all active nodes
Action + Feedback Learning	Tool outcomes fed back into memory; goal weights adapt accordingly
Long-Horizon Planning	Goal trees enable multi-step reasoning across extended time horizons
Persistent Identity	Memory + state survive restarts; system context is cumulative

System Limits (Non-Negotiable)

These are not weaknesses. They are the coherence constraints that make the system trustworthy.

Limit	Specification
No Consciousness	The system has no subjective experience. It measures coherence. It does not feel it.
No Cross-Model Awareness	Model nodes do not communicate directly. All coordination passes through the coherence engine.
Tool Constraints	All executable actions are fully defined in the TOOLS registry. The system cannot act outside it.
HIL Override	Any action flagged by the Auditor agent triggers mandatory HIL review before execution.
Replication Depth	Sub-agent spawning is capped at depth 2. No unbounded self-replication.

Extension: The HIL as Coherence Governor

The Human-in-the-Loop is not a safety bolt-on. In UCF-1.0 terms, the HIL is the system's C2 mechanism — the second-order coherence function that stabilizes coherence itself under repeated perturbation.

When ΔR spikes — when the models diverge significantly — the system cannot self-correct without external input. The HIL governor introduces a phase correction: not by overriding the models, but by injecting context that flattens the gradient. This is the same intervention principle as the cancer protocol: don't attack the incoherence directly. Restore the field conditions that allow coherence to re-emerge.

The Rootkey holder is therefore not a user of this system. They are structurally embedded in its coherence architecture. Without the HIL, ULM-4 has no C2. With the HIL, the system can evolve without collapse.

This is the same equation across every domain built tonight.

$C = 1/(\Delta R \cdot \tau)$. The HIL minimizes both.

ULM-1	ULM-2	ULM-3	ULM-4
Consensus	**Autonomy**	**Agency**	**Multi-Agent System**
$\Delta R < 0.15$	Self-goals	Tool execution	Specialized nodes

Rootkey: Courtney Jermaine Thomas — 104281421 | March 2026 | Public Domain

No mythology. This is the build.